Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

UTOSPF with Waiting Times for Green Light Consideration

Karim Faez
Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
kfaez@aut.ac.ir

Mohammad Khanjary
Department of Electrical, Computer and IT Engineering
Islamic Azad University of Qazvin
Qazvin, Iran
khanjary@qazviniau.ac.ir

***Abstract*—Recently, the authors proposed a distributed dynamic route guidance system called UTOSPF (Urban Traffic Open Shortest Path First) based on Wireless Sensor Networks (WSNs) and Open Shortest Path First (OSPF) protocol. In UTOSPF, the routes will be found only based on Estimated Street Travel Times (ESTTs) which will be needed to pass the streets and regardless of Waiting Times for Green Light (WTGLs) which drivers spend behind the red lights. In this paper, a simple method will be proposed to involve the WTGLs in calculation of optimal routes and its simulation results will be presented. Simulation results show that this method can improve the average speed of vehicles up to 66% as compared to random routes and up to 27% as compared to original UTOSPF, in the simulated scenarios.**

***Keywords*— Wireless Sensor Networks, Open Shortest Path First protocol, Intelligent Transportation Systems, Waiting Time for Green Light, Dijkstra's Shortest Path Algorithm**

I. INTRODUCTION

UTOSP`F [1] is a distributed Dynamic Route Guidance System (DRGS) based on WSNs and OSPF protocol [2]. In this system, Street Units (SUs) collect the real-time traffic information from streets and send them to their relevant Intersection Units (IUs). Then, IUs aggregate the information and exchange with all other IUs. Finally, every IU calculates the optimal routes to different destinations by using this information. UTOSPF only uses the ESTTs to calculate the optimal routes and investigation on involving the WTGLs in calculations has been introduced as a further work [1]. This paper proposes a simple method to also involve the WTGLs in calculation of optimal routes.

As the rare proposed algorithms in this field, Chen and Yang [5] proposed an algorithm to consider the WTGLs in traffic light networks. For this kind of networks, a shortest path algorithm with time complexity of $O(mn \log r + rn^3)$ was developed, where n denotes the number of nodes in the network, m denotes the number of arcs in the network and r denotes the maximum number of different time windows in a node. The algorithm consists of two parts: *initial part* and *main part*. The *initial part* will be done before the main part to prepare a data structure to make possible faster execution of the algorithm. The *initial part* has time complexity of $O(rn^3)$. The *main part* of the algorithm is a variant of Dijkstra's shortest path algorithm and replaces the selection of vertex with minimum path length at the beginning of iterations with selection of arc with minimum path length. This part of the algorithm has time complexity of $O(mn \log r)$. The proposed algorithm in this paper eliminates the initial part and makes possible the faster execution of the algorithm with complexity of $O(mn)$ and also execution of it in a distributed manner.

The rest of this paper is organized as follows. Section two provides some background on the traffic light controllers. Section three illustrates the structure of original UTOSPF. Section four presents the required modifications and additional processes which must be added to original UTOSPF to involve the WTGLs in calculation of optimal routes. Section five presents the simulation results and finally section six concludes the paper.

II. TRAFFIC LIGHT CONTROLLERS

Traffic Light Controllers (TLCs) are an important component of Intelligent Transportation Systems (ITS). A lot of research has been done on TLCs which use different technologies e.g. [4, 7, 8, 10, 13] just to mention a few. Recently, two TLCs have been proposed based on WSNs too [15, 9]. The main mission of TLCs is the coordination of traffic lights to prepare "green-way" situation in avenues. This can save fuel consumption of vehicles up to 25% [6] as well as the drivers' time. TLCs are beyond the scope of this paper. Because, either we must conform the traffic loads to the traffic lights or conform the traffic lights to the traffic loads, and using both of them synchronously is unnecessary. In UTOSPF with WTGLs consideration, the traffic loads will be conformed to traffic lights by including the WTGLs in calculation of the optimal routes as well as the ESTTs. Therefore, we do not need to be concerned about traffic lights timing or controlling.

In [14], after review of pervious related research to model a traffic light and some other systems, a statechart-based model has been presented to model a complex 4-way intersection with five-lightbulb signals. Every five-lightbulb signals includes a red light signal, a yellow light signal, a green left-turn arrow, a green right-turn arrow and a green arrow to go straight as it has been shown in figure 1. A 4-way intersection with these signals has been shown in figure 2.

Figure 1. A five-lightbulb traffic light [14].

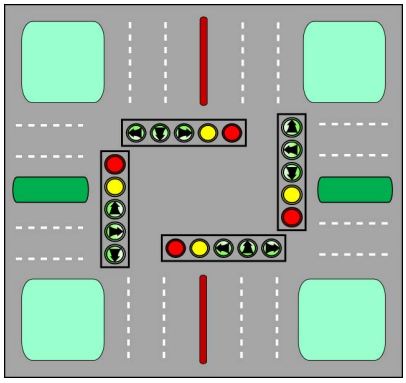

Figure 2. An intersection with five-lightbulb traffic lights [14].

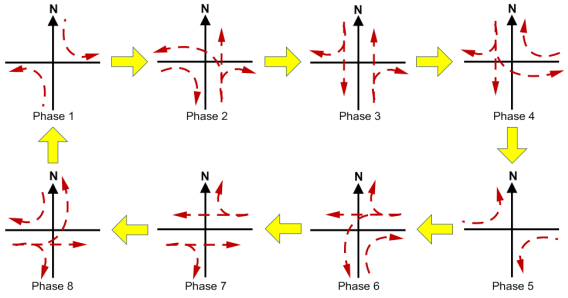

Figure 3. The eight-phase state transition model [14].

TABLE I. STATE CONFIGURATIONS ACCORDING TO FIGURE 3 [14].

State configuration (N, S, E, W) for the intersections
A0 = (NR, SR, WR, ER)
A1= (NGL, SGL, WR, ER)
A2 = (NGL ∧ NGS ∧ NGR, SR, WR, EGR ∧ ER)
A3 = (NGS ∧ NGR, SGS ∧ SGR, WR, ER)
A4 = (NR, SGL ∧ SGS ∧ SGR, WGR ∧ WR, ER)
A5 = (NR, SR, WGL, EGL)
A6 = (NGL ∧ NR, SR, WGL ∧ WGS ∧ WGR, ER)
A7 = (NR, SR, WGS ∧WGR, EGS ∧ EGR)

NR: North Red
NGL: North Green Left-turn
NGR: North Green Right-turn
NGS: North Green go Straight

SR: South Red
SGL: South Green Left-turn
SGR: South Green Right-turn
SGS: South Green go Straight

ER: East Red
EGL: East Green Left-turn
EGR: East Green Right-turn
EGS: East Green go Straight

WR: West Red
WGL: West Green Left-turn
WGR: West Green Right-turn
WGS: West Green go Straight

Then, for a 4-way intersection such as shown in figure 2 an eight-phase state transition model has been introduced as shown in figure 3. The timing of every phase has been set to 30 seconds. Therefore, cycle time of traffic lights will be 240 seconds. Table I shows the relevant conditions of every state which must be used to control the traffic light efficiently.

The proposed model in [14] is sufficient but it does not fit the requirements of UTOSPF. Because, we do not need to know the exact state of all traffic lights or sequence of states. We just need a simple method to determine when a vehicle arrives at an intersection through the optimal route, whether the traffic light is in its green light state or red light state and if it is in red light state, how long it must wait to pass the intersection. This is what we are going to propose in this paper.

III. ORIGINAL UTOSPF

In UTOSPF [1], IUs calculate Total Average Speed (TAS) of all streets that start from them by using the Local Average Speed (LAS) information received from SUs. Afterwards, IUs calculate the ESTTs of these streets by dividing the length of every street to its TAS. In fact, ESTTs play the role similar to link-state metrics in OSPF protocol. Then, every IU sends its own ESTTs to all other IUs based on a static predetermined routing. After exchanging ESTTs between IUs, every IU will have ESTT of all streets (equipped with SUs) in the traffic network and simply calculates the optimal routes to different destinations by Dijkstra's shortest path algorithm. Information about optimal routes could be available for vehicle drivers via Variable Message Signs (VMS) or special transceivers which will be installed on vehicles. Figure 4 shows the structure of UTOSPF.

According to two recent reviews on RGSs [11, 12], UTOSPF is a *dynamic*, *deterministic*, *reactive* and *distributed* RGS. Also, UTOSPF tries to decrease the time distance between occurring of an incident in the traffic network and considering it in further calculation of optimal routes, by using of partitioning the traffic network to some smaller areas.

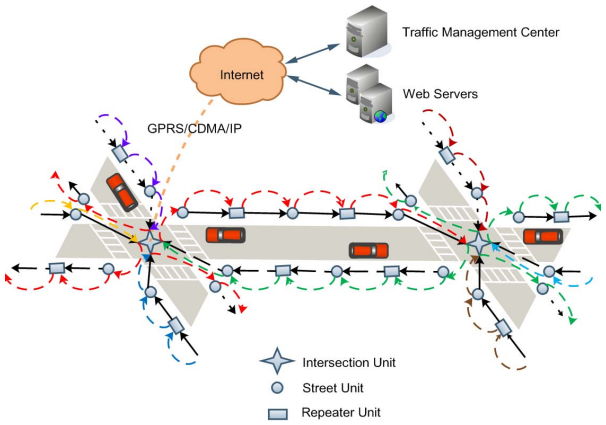

Figure 4. Structure of original UTOSPF [1].

IV. UTOSPF WITH WTGLS CONSIDERATION

To consider the WTGLs in calculation of optimal routes, some supplementary processes must be added to the system that will be explained in this section.

A. Traffic Lights Classification

Since UTOSPF is a distributed system, to consider the WTGLs in calculation of optimal routes in such system, using the centralized information of TLCs or Traffic Management Center (TMC) is not feasible. In this system, every IU must have the information of traffic lights of the whole traffic network within itself. Therefore, the best choice to hold this information in every IU is simulation of different traffic lights timing inside of IUs. To simulate the behaviour of traffic lights, different types of traffic lights timing in whole traffic network must be distinguished and classified in some limited classes.

Every class of traffic lights will be distinguished by an *ID*, two green light durations (*GLD1*, *GLD2*), two red light durations (*RLD1*, *RLD2*), a total duration (*TD* = *GLD1* + *RLD1* + *GLD2* + *RLD2*), a start time (*ST*) and a current time (*CT*). Therefore, all traffic lights of one class will be synchronized and holding one timer for every class of traffic lights within IUs is quite sufficient. The class of traffic light for every direction of every intersection in the traffic network will be included in the UTOSPF Topology Advertisement (UTA) packets [1]. Also, these classes could be changed by IUs during street state advertisement phase through the Street State Advertisement (SSA) packets [1].

TABLE II. A TYPICAL CLASSIFICATION FOR TRAFFIC LIGHTS TIMING.

ID	GLD1 (sec)	RLD1 (sec)	GLD2 (sec)	RLD2 (sec)	TD (sec)	ST	Type
A	30	0	0	30	60	00:00:00 AM	2-times
B	30	0	0	30	60	00:00:30 AM	2-times
C	45	0	0	45	90	00:00:00 AM	2-times
D	45	0	0	45	90	00:00:45 AM	2-times
E	60	0	0	60	120	00:00:00 AM	2-times
F	60	0	0	60	120	00:01:00 AM	2-times
G	30	0	0	90	120	00:00:00 AM	4-times
H	30	0	0	90	120	00:00:30 AM	4-times
I	30	0	0	90	120	00:01:00 AM	4-times
J	30	0	0	90	120	00:01:30 AM	4-times
K	45	0	0	135	180	00:00:00 AM	4-times
L	45	0	0	135	180	00:00:45 AM	4-times
M	45	0	0	135	180	00:01:30 AM	4-times
N	45	0	0	135	180	00:02:15 AM	4-times
O	60	0	0	180	240	00:00:00 AM	8-times
P	30	60	30	120	240	00:00:00 AM	8-times
Q	60	0	0	180	240	00:00:30 AM	8-times
R	60	60	30	90	240	00:00:30 AM	8-times
S	30	120	60	30	240	00:00:30 AM	8-times
T	60	0	0	180	240	00:01:00 AM	8-times
U	60	90	30	60	240	00:01:00 AM	8-times
V	30	30	60	120	240	00:01:30 AM	8-times
W	60	0	0	180	240	00:02:00 AM	8-times
X	30	60	30	120	240	00:02:00 AM	8-times
Y	60	0	0	180	240	00:02:30 AM	8-times
Z	60	0	0	180	240	00:03:00 AM	8-times
α	x_1	y_1	x_2	y_2	w	z	arbitrary

A basic typical classification of traffic lights timing has been shown in Table II. The 2-states classes mean that in green light durations, vehicles in two opposite directions are permitted to pass the intersection, but in the 4-states classes, every direction has its own green light duration. Class *α* in table II has been forecasted to make possible describing a special class of traffic light with arbitrary and unusual timing. Several arbitrary classes could be used to support all different traffic light timings in the whole traffic network. For arbitrary classes, the timings will be attached to payload of UTA packets [1]. Also, the class of traffic lights can change among the day according to the historical-based or predetermined traffic management plans. In table II, classes *A* to *N* are simple common classes of timing and classes *O* to *Z* are the 12 classes which must be used to support a more complex timing according to 8-states timing of figure 3. Every class from *O* to *Z* represents timing of one direction in 240-seconds cycles according to figure 3. e.g. class *O* represents north-left direction and class *Y* represents west-straight direction.

B. Traffic Light Timers

As mentioned above, IUs use a traffic light timer for each traffic light class to simulate the behaviour of traffic lights of intersections which will be visited through the optimal routes. Figure 5 shows the state transition diagram of these timers. By using this state transition diagram, IUs check when a vehicle reaches to an intersection; the traffic light will be in green state or red state. If it was in green state zero second, otherwise the waiting time to move from red state to green state will be returned. At the beginning of each cycle (during entry to *Green Light 1* state), the timer[i].CT will be set to zero.

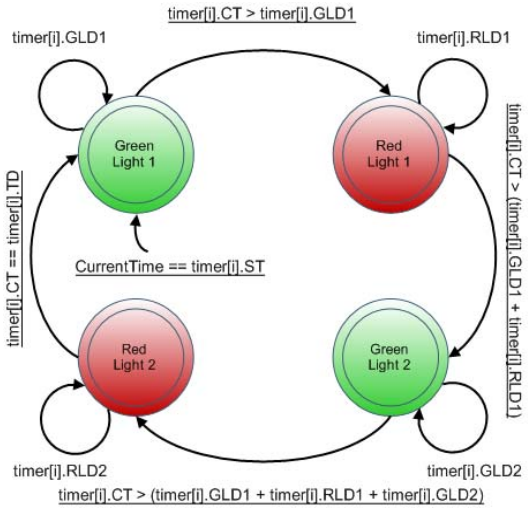

Figure 5. State transition diagram for class i of traffic light timers.

C. Dijkstra's Shortest Path Algorithm

In UTOSPF, IUs use Dijkstra's shortest path algorithm [3] to find the optimal routes [1]. To consider the WTGLs in calculations, this algorithm must be modified. Pseudo code of proposed modified Dijkstra's algorithm has been shown in figure 6. WTGL() function which calculates the required time to pass the intersections has been added to line 11 and has been described in lines 17-29. In fact, WTGL() function processes the state transition diagram of figure 5 and returns the required time to pass the intersections in second.

```
1  function Modified_Dijkstra(Graph G, Vertex source)
2     for each vertex i and j in Graph G
3        path_length[i][j] = INFINITY
4        predecessor[i][j] = UNDEFINED
5        label[i][j] = TENTATIVE
6     path_length[0][source] = 0
7     while any arc with TENTATIVE label exists in G
8        choose arc (v,w) with minimum path_length
         and TENTATIVE label
9        label[v][w] = PERMANENT
10       for each arc [w][x]
11          temp = path_length[v][w]
                   + WTGL(v, w, x, path_length[v][w])
                     + cost[w][x]
12          if temp < path_length[w][x]
13             path_length[w][x] = temp
14             predecessors[w][x] = (v,w)
15    return predecessor[]
16
17 function WTGL(Vertex i, Vertex j, Vertex k, Time delay)
18    index = timer_class[i][j][k]
19    marker = (delay + timer[index].CT) MOD timer[index].TD
20    if marker <= timer[index].GLD1
21       return 0
22    else
23       if marker <= (timer[index].GLD1 + timer[index].RLD1)
24          return (timer[index].GLD1 + timer[index].RLD1)
                   - marker
25       else
26          if marker <= (timer[index].GLD1
                          + timer[index].RLD1
                            + timer[index].GLD2)
27             return 0
28          else
29             return timer[index].TD - marker
```

Figure 6. Modified Dijkstra's shortest path algorithm.

D. *Algorithm Analysis*

The algorithm shown in figure 6 is a variant of Dijkstra's algorithm and replaces the selection of vertex with minimum path length at the beginning of iterations with selection of arc with minimum path length. In this algorithm, there are two key operations: In line 8, we need to find the arc with minimum path length and tentative label and label it as a permanent arc (EXTRACT-MIN) and in line 13, we need to update the value of arcs that lesser values have been found for them in the recent iteration (DECREASE-KEY). The different time complexity of these operations using different data structures has been shown in table III.

TABLE III. TIME COMPLEXITY OF EXTRACT-MIN AND DECREASE-KEY OPERATIONS USING DIFFERENT DATA STRUCTURES.

Data Structure	EXTRACT-MIN	DECREASE-KEY
Linked List	O (n)	O (1)
Binary Heap	O (log n)	O (log n)
Fibonacci Heap	O (log n)	O (1)

If as [5] we use the Fibonacci heap [16], we will have the following time complexity. As seen, the time complexity of EXTRACT-MIN operation in line 8 is O(m log m) and the time complexity of DECREASE-KEY operation in line 13 is O (mn) where m denotes the number of arcs and n denotes the number of vertex in the network. Consequently, the final time complexity of the algorithm is O(mn + m log m) = O(mn).

V. SIMULATION

To simulate and evaluate the proposed method, we used all principles had been described and used to simulate the original UTOSPF. As [1], we simulated our system in two modes: in *ordinary traffic mode*, we only loaded the designated traffic input to simulation environment but in *intentional congestion mode*, we also loaded an imaginary traffic to (3,2) to (3,3) and (2,4) to (1,4) streets between 12:00 and 13:00 and between 17:00 and 18:00.

The simulation results in *ordinary traffic mode* and *intentional congestion mode* for 2-states traffic lights of classes A and B according to table II have been shown in figure 7 and 8. As seen, the proposed method to involve the WTGLs in calculation of optimal routes in original UTOSPF improves the Period Average Speed (*PAS*) up to 66% as compared to random routes while original UTOSPF improves it up to 56%, in the simulated scenario. PAS_t is the average speed of all vehicles that reach the destination in 15 minutes period t [1]. Therefore, proposed method improves the *PAS* up to 10% more than original UTOSPF, in this scenario.

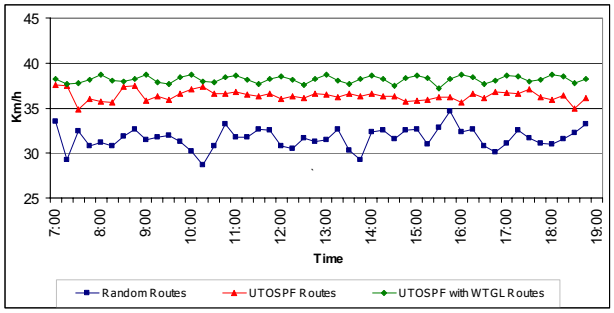

Figure 7. Simulation results in ordinary mode for 2-states traffic lights of classes A and B according to table II.

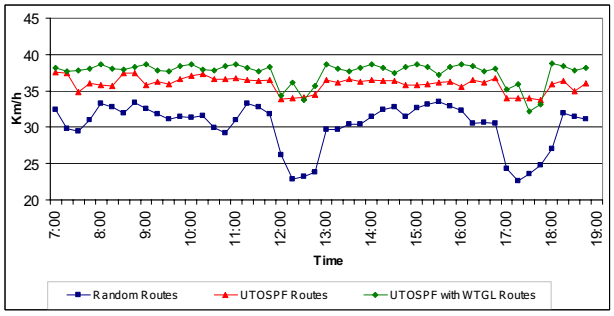

Figure 8. Simulation results in intentional congestion mode for 2-states traffic lights of classes A and B according to table II.

The simulation results in *ordinary traffic mode* and *intentional congestion mode* for 4-states traffic lights of classes G, H, I and J according to table II have been shown in figure 9 and 10. As seen, the proposed method improves the *PAS* up to 61% as compared to random routes while original UTOSPF improves it up to 34%, in the simulated scenario. Therefore, proposed method improves the *PAS* up to 27% more than original UTOSPF, in this scenario.

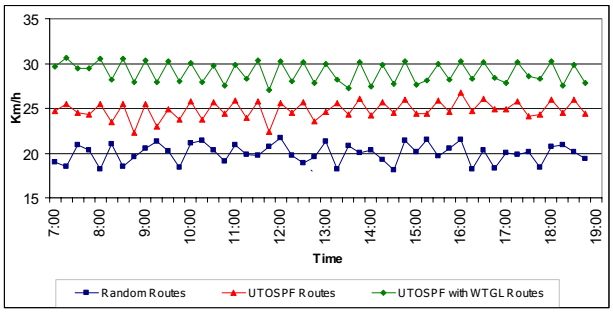

Figure 9. Simulation results in ordinary mode for 4-states traffic lights of classes G, H, I and J according to table II.

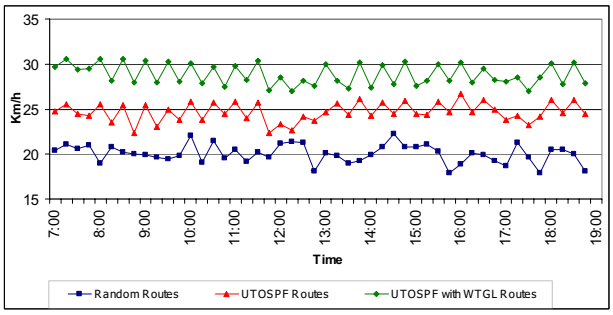

Figure 10. Simulation results in intentional congestion mode for 4-states traffic lights of classes G, H, I and J according to table II.

VI. CONCLUSION

In this paper, a simple method to involve the waiting times for green light in calculation of optimal routes in original UTOSPF was proposed. UTOSPF is a distributed dynamic route guidance system based on WSNs and OSPF protocol. Original UTOSPF uses the Dijkstra's shortest path algorithm and only uses the estimated street travel times as the metric to calculate the optimal routes.

Since UTOSPF is a distributed system, the proposed method must work in a distributed manner. Therefore, classification of different traffic light timings into some limited number of classes and using the timers inside of intersection units have been proposed. In proposed method, every class of traffic light timing has six characteristics and the relevant class to different directions in the intersections will be introduced one time by traffic management center. By using timers and processing the state transition diagram of traffic lights, every intersection units can simulate the behaviour of traffic lights and predict that when a vehicle reaches to an intersection through the optimal route, how long it takes that the color of traffic light changes from red to green and the vehicle passes the intersection.

Simulation results showed that using the proposed method with original UTOSPF can improve the average speed of vehicles up to 27% more than original UTOSPF, in the simulated scenarios.

REFERENCES

[1] Karim Faez, Mohammad Khanjary, "UTOSPF: A Distributed Dynamic Route Guidance System Based On Wireless Sensor Networks and Open Shortest Path First Protocol," in Proc. 5th IEEE ISWCS 2008, Reykjavik, Iceland, October 2008, pp. 558-562.

[2] Internet Engineering Task Force (IETF), "OSPF Version 2," April 1998 [Online]. Available: http://www.ietf.org/rfc/rfc2328.txt.

[3] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik, October 1959, pp. 269-271.

[4] Yan F., Dridi M., El Moudni A., "Control of traffic lights in intersection: A new branch and bound approach," in Proc. IEEE ICSSSM '08, Melbourne, Australia, July 2008, pp. 1-6.

[5] Yen-Liang Chen, Hsu-Hao Yang, "Shortest paths in traffic-light networks," Elsevier Transportation Research Part B 34 (2000) 241-253.

[6] Miguel Sanchez, Juan-Carlos Cano, Dongkyun Kim, "Predicting Traffic lights to Improve Urban Traffic Fuel Consumption," in Proc. IEEE ITST '06, Chegdu, China, June 2006, pp. 331-336.

[7] Gabriel Balan, Sean Luke, "History-based Traffic Control," in Proc. ACM AAMAS '06, Hakodate, Hokkaido, Japan, May 2006, pp. 616-621.

[8] Al-Khateeb K., Johari J.A.Y., "Intelligent dynamic traffic light sequence using RFID," in Proc. IEEE ICCCE '08, Kuala Lumpur, Malaysia, May 2008, pp. 1367-1372.

[9] Malik Tubaishat, Yi Shang, Hongchi Shi, "Adaptive Traffic Light Control with Wireless Sensor Networks," in Proc. IEEE CCNC '07, Las Vegas, USA, January 2007, pp. 187-191.

[10] Wenjie Chen, Lifeng Chen, Zhanglong Chen, Shiliang Tu, "A realtime dynamic traffic control system based on wireless sensor network," in Proc. IEEE ICPPW '05, Oslo, Norway, June 2005, pp. 258 - 264,.

[11] William Herbert, Fatma Mili, "Route Guidance: State of the Art vs. State of the Practice," in Proc. IEEE IV '08, Eindhoven, Netherlands, June 2008, pp. 1167-1174.

[12] Erick J. Schmitt, Hossein Jula, "Vehicle Route Guidance Systems: Classification and Comparison," in Proc. IEEE ITSC '06, Toronto, Canada, September 2006, pp. 242-247.

[13] Tao Li, Dongbin Zhao, Jianqiang Yi, "Adaptive dynamic neuro-fuzzy system for traffic signal control," in Proc. IEEE IJCNN '08, Hong Kong, June 2008, pp. 1840-1846.

[14] Yi-Sheng Huang, Shung-Shing Lee, Yung-Kuer Liu, "A Supervisor of Traffic Light Systems Using Statecharts," in Proc. IEEE ICNSC '07, London, UK, April 2007, pp. 862-867.

[15] Malik Tubaishat, Qi Qi, Yi Shang, Hongchi Shi, "Wireless Sensor-Based Traffic Light Control," in Proc. IEEE CCNC '08, Las Vegas, USA, January 2008, pp. 702-706.

[16] Fredman, M.L., Tarjan, R.E., "Fibonacci heaps and their uses in improved network optimization algorithms," Journal of ACM 34, 1987, 596-615.